A Protocol Processing Architecture Backing TCP/IP-based Security Applications in High Speed Networks

Christos Georgopoulos[1], George Konstantoulakis[1], Theofanis Orphanoudakis[1], Nikos Nikolaou[2], Jorge-A. Sanchez-P. [2], Nikos Mouratidis[1], Kostas Pramataris[2] and Nick Zervos[1]

[1] Ellemedia Technologies, Syggrou Av. 223, 17121 Athens, Greece
{cgeorg, gkonst, fanis, nmo, nzervos}@ellemedia.com

[2] Lucent Technologies, Forward Looking Work EMEA, Botterstraat 45, P.O.Box 18, 1270 AA Huizen, The Netherlands
{nikolaou, jsancheze, kpram}@lucent.com

Abstract. In this paper, the architecture, system modules and functional design of a reconfigurable protocol processor, developed under the PRO^3 project [1], are presented. The protocol processor aims in accelerating execution of telecom protocols by extending a high-performance RISC core with reconfigurable pipelined hardware. CPU demanding and (hard) real-time protocol functions will be handled by the programmable hardware, while the remaining functions as well as higher layer protocols will be handled by the on-chip RISC in an integrated way. The paper focuses on a firewall application that exploits the PRO^3 architecture and is under development in the project. A firewall guarantees the security and privacy of data transactions of networking applications.

1 Introduction

The trend of data, voice, and video traffic convergence and the tremendous growth in data traffic, particularly that which is associated with the Internet, has prompted many discussions on the hardware and software needed for the future. Newer bandwidth-eager end-user software applications and faster processors in desktop and server systems are placing enormous demands on the current networking architecture. As a result, the network is evolving into a highly complex and sophisticated environment: networking bandwidth continues to double every four months; guaranteed quality and priority customization is anticipated to all data, voice and video applications; networks will be running packets; software will be the key in making networks work.

[1] This paper describes work undertaken in the context of the IST-1999-11419 Protocol Processor Project (PRO^3) [5], a 2 ½ years research and development project. The IST programme is partially funded by the Commission of the European Union. The authors would like to acknowledge the contributions of their colleagues from Lucent Technologies, Hyperstone electronics, IMEC, National Technical University of Athens and Ellemedia Technologies.

The rapid growth in the dimension of networks, along with the always increasing users' demands for networking services, have imposed the development and deployment of high-capacity telecommunication systems (carrier and ISP backbones running OC3, OC12, OC-48 and some OC-192 speeds). Such systems involve modules of high throughput, which have their time critical functions realized in application specific standard products. The concept of improving the computation performance of protocols by mapping the time-critical (wire speed) functionality onto hardware structures, originated during the evolution of ATM. Currently, the trend is to apply this concept to routers, servers, multi-layer switches and VoIP gateways required to support efficiently multiple links at gigabit rates.

The power required for the processing of protocol functions at wire speed is usually obtained either by generic microprocessors that are designed with the flexibility to perform a variety of functions, but at a slower speed or Application Specific Integrated Circuits (ASIC's) that are designed to meet a specific functional requirement. In the new demanding environment ASIC's are extremely efficient within their specific set of tasks but very difficult, if not impossible, to change once they have been designed, and cannot be modified with a simple software upgrade. With time-to-market becoming the dominant force in the networking world (for an ASIC it can be up to eighteen months), many companies have turned to Reduced Instruction Set Computing (RISC) technology which embeds small sets of instructions within a processor to make the processor very flexible. RISC's can execute their code very fast due to instructions simplicity, and require fewer transistors, which in turn makes them cheaper to design and produce, but since the decisions are made in software, the RISC is much slower than the ASIC. Another option is a hybrid approach. This combines both chip technologies, using a RISC processor as the central core, and ASIC's to perform the specific tasks. These components called Network Processors have exhibited an enormous advance in the turn of the millennium [6, 7, 8, 9].

As it concerns higher layer protocol functions that are not performed at wire speed, such as: routing protocols, statistical compiling and reporting, error processing, connection admission control, and traffic and resource management, often today, more than one high performance processing unit is employed. In such systems, the processing units are inadequate in supporting the protocol processing requirements for the entire set of active sessions. This constitutes a major system resource bottleneck, because the complexity of the protocol algorithms requires higher computational power than that offered by today's processor technology [1, 2, 3, 4]. An example illustrating this situation is the ATM network, where each switch is required to process an active signalling stack (with two transport protocol stacks instances – SAR 5, CPCS, SSCOP and one signalling protocol instance – Q.2391) per user connection. Hybrid components will be required also in this case in the near future.

The PRO^3 system architecture, presented in section 2, is a hybrid approach to the demanding protocol processing puzzle. The PRO^3 system is enhanced with hardwired functionality devoted to speed up streaming and networking operations. The reconfigurable stages benefit the use of a RISC processor when executing more than one protocols concurrently. The RISC core accomplishes the task of maintaining the higher protocol levels and operation of these modules in parallel. The Protocol Processor is considered as the next step in the evolution path of communication processor technology and constitutes a new concept in specialized processing elements (such as

DSP). Section 3 analyses the functional architecture of the system and presents the main modules of the system. Section 4 matches the firewall application to the PRO^3 architecture and discusses implementation issues. Finally, Section 5 concludes the paper and presents the implementation plan.

2 PRO^3 System Architecture

PRO^3 targets the tight coupling of software and hardware for the efficient execution of telecommunication protocols in embedded, programmable architectures. The project will design, develop and fabricate a versatile protocol processor to accelerate execution of telecom and data transport protocols by extending a high-performance RISC core with reconfigurable pipelined hardware. CPU demanding and (hard) real-time protocol functions will be handled by the programmable hardware, while the remaining functions as well as higher layer protocols will be handled by the on-chip RISC in an integrated way. Applications that utilize the PRO^3 processor will be developed to demonstrate the enhanced capabilities.

PRO^3 introduces a new concept in processors: integrating high and low level protocol processing units optimized to perform in parallel. This integration involves an innovative scheme with interconnection between a RISC and a Reconfigurable Pipelined Module (RPM). The resulting processor architecture will be tailored to considerably accelerate the performance of protocols and streaming processes. The RPM which will be implemented either in function specific units or on an array of a bit-sliced processor such as a Field Programmable Gate Array (FPGA), will be used for the realization of the selected protocol functions. Additional modules will be integrated around the enhanced core including the timer events generator, the packet level memory management and the generic coder/decoder.

The component will efficiently realize the set of the protocol functions performed most often as well as the low level operating system functions that support the execution of protocols including timers, inter-process communications and memory control. Analysis of protocol performance within such systems shows that a small specific subset of the protocol functions (i.e. less than 10%) is active during large periods of time (i.e. more than 95% of the entire time span). The other protocol functionality is active when errors occur or for set-up and tear down of a protocol session. PRO^3 targets at realizing the aforementioned specific set of the 10% of the protocol functions in the reconfigurable module and the remaining functions in the RISC core. The key idea is not to perform the entire protocol on the fly (since it is very expensive) but to accelerate it considerably. The following is a list of PRO^3 innovative aspects:

- Integration of a state-of-the-art RISC with a reconfigurable module able to deliver the needed processing power to support efficiently many thousands of (different) protocol instances.
- Introduction of message recognition information coming from the input module in order to reconfigure RPM. Inclusion of logic for fast lookup tables in hardware and high-speed interfaces.

- Provision of a pool of timer resources for the realization of the (many) thousands of active watch-dog timers. Since it must be able to support thousands of active instances of protocol FSMs, special circuitry should be designed.
- Provision of generic coding-decoding functions for the bit or byte level handling of protocol streams. Each protocol has unique and distinct coding-decoding rules at its lower interface. A protocol receives a byte stream from its lower interface and gives another byte stream at its upper interface. In order for a protocol execution machine to extract the information fields, the received stream is usually parsed at bit level.
- Memory management features for the real time context switching, memory allocation and stream buffering (for segmentation/reassembly).
- Realization of high performance and demanding applications that efficiently utilize the PRO^3 system.

In summary, the PRO^3 architecture involves an independent pipelined data path (protocol messages and data pass through) and a control interface (realized as a simple master/slave processor bus) to fetch software to the RISC/DSP and communicate with the host system. Special attention in the proposed design is given to the efficient support of multiple instances of the same protocol stack by the implemented logic, as well as the ability to improve on the performance of a diversity of CPU demanding network protocols.

3 PRO^3 Functional Design and System Modules

The functional architecture of the PRO^3 system is depicted in Fig.1. The component consists of a central processing unit (the RISC core and the RPM) as well as a set of on-chip peripherals, common to protocols and streaming tasks. Thus, the same component with different configuration will be able to realize many different protocol Finite State Machines (FSMs) that require high performance execution and handling of messages with low propagation/processing delay. The feedback bus returns messages to the input of the component in case a multi-protocol stack is implemented. Certain priority rules will be applied along with specific access to/from the system.

The component, depicted in Fig.1, has a pipelined data path where the high bandwidth streaming messages are injected (and being processed) and a control interface for software fetching. The following modules constitute peripherals of the protocol processor:

- The input/output module of the system is a general-purpose parallel peripheral interface. The actual load of this interface depends on the application.
- Message Recognition: The main function of this module is to recognize the incoming message and to assign an internal handler for subsequent processing. The handler is important not only for PRO^3, but can also be sent to the application to aid further processing/classification of higher layer messages. As presented in Figure 1, a direct communication path is foreseen from the Message Recognition block to the mixed core. This signal will inform in advance the core on the message to be

processed so that, if needed, the RISC will be able to reconfigure the entire reconfigurable module or part of it.

- Generic encoder/decoder: This module acts both at the receiver and the transmitter side of PRO^3. On the receiver, it decodes the incoming byte stream and extracts the respective protocol information fields. The parsing rules depend on the realized protocol and will be based on simple description commands. For simplicity the module may apply only to the high speed and low propagation delay messages that are processed by the RPM and not to those forwarded to the RISC core.
- Timers pool: Time-out timers and watch dog timers are an important part in implementation since they generally run independently of the protocol FSM and message reception flow. Having in mind that each protocol instance needs at least a few timers (of different time scales), a high end system supporting thousands of protocol instances, spends considerable part of its processing power only to keep track of time for all of them. An efficient method based on a virtual clock queue mechanism is being studied to maintain time tracking and produce time-out events.
- Memory management: The module assigns memory blocks of variable size to each protocol in order to store either data structures, received messages or parts of payloads in case of higher layer message re-assembly. Furthermore, efficient memory management is needed for fast context switching of protocol FSMs, FIFOs creation, etc. The memory management and control module will realize some low-level functions in hardware that are provided typically by operating systems. The module buffer will be shared to minimize resources and costs. For cost and scalability reasons the buffer will be external to the system using cost effective DRAM whilst the logic for memory management will be internal.

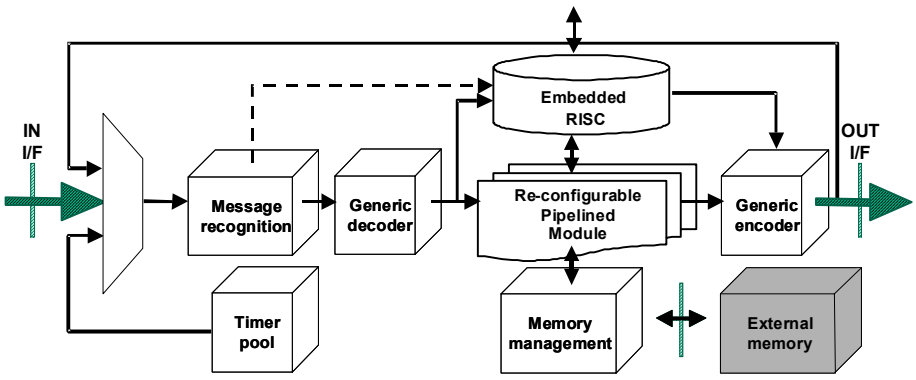

Fig. 1. PRO^3 functional architecture and data path

4 A Firewall Application implemented on the PRO^3 Architecture

The PRO^3 system can be utilized for the efficient implementation of various networking applications like processing of lower layer protocols (synchronization, arbitrary length field extraction, CRC calculation, scrambling/descrambling), processing of

higher layer protocols (PDU reassembly, queue management, arbitrary length field extraction, CRC/checksum calculation, connection state handling, timer management), packet routing, packet switching, QoS provision, security, etc.

Security in TCP/IP protocols is one of the applications that will be demonstrated in the PRO^3 project. Security might include functions like packet classification, flow classification and connection state handling (state-full inspection), higher layer PDU reassembly (application level firewalls, packet decryption), encryption/decryption.

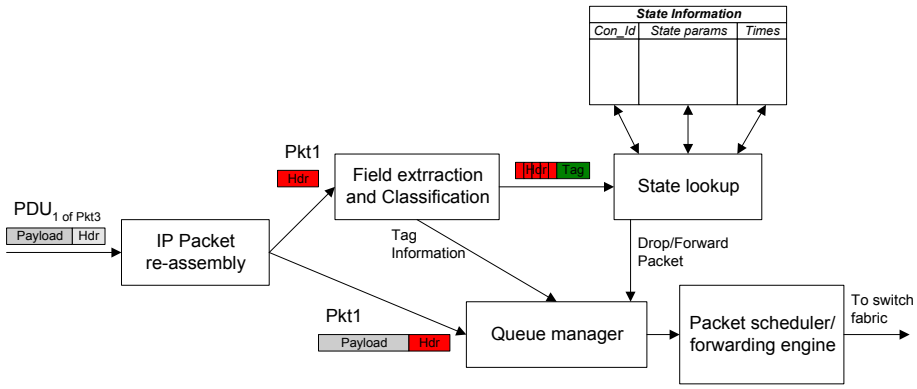

Fig. 2. Mapping of the firewall application to the PRO^3 architecture

A firewall system guarantees the security and privacy of data transactions of networking applications. It can be implemented either in a centralized architecture, where a single router filters all packets, or in a distributed manner, consisting of a combination of technologies in routers and hosts. A firewall system enforces a so-called access control policy between two networks. All the incoming and outgoing traffic is destined through the firewall and only authorized traffic, as defined by the local security policy, is allowed to pass through.

Firewalls are widely used to give users access to the Internet in a secure fashion as well as to separate a company's public servers (e.g. Web server) from its internal network. They are also used to keep internal network segments secure. To reach control decisions for TCP/IP based services (e.g. whether to accept, reject, authenticate, encrypt and/or log communication attempts) a firewall must obtain, store, and manipulate information derived from all communication layers.

However, isolation and examination of packets independently is not a sufficient method to provide security. State information, derived either from the near past of a communication session or from other applications run in the past, is an essential factor in making the control decision for new communication attempts. Depending upon the communication attempt, both the communication state (derived from past communications) and the application state (derived from other applications) may be critical in the control decision. Thus, to ensure the highest level of security, a firewall must be capable of accessing, analyzing and utilizing the following:

- Communication information extracted from each one of the seven OSI layers.

- Communication-derived state representing the state derived from the past behavior of the communication (e.g., the outgoing PORT command of an FTP session could be saved so that an incoming FTP data connection can be verified against it).
- Application-derived state signifying the state information extracted from applications that have already been checked (e.g., a previously authenticated user would be allowed access through the firewall for authorized services only).
- Information manipulation, which enables the evaluation of flexible expressions, based on all the above factors.

To this respect, stateful inspection is able to meet all the security requirements defined above, while traditional firewall technologies, such as packet filters and application-layer gateways, fail in some areas. Stateful inspection mainly takes place at the LAN level, where the firewall acts as the bridge/gateway between a LAN and the Internet (or even between segments of the same Intranet).

The proposed application for the PRO^3 system is a migration from the usual implementation of firewalls, which are built around a commercial general-purpose processor (both for firewall appliances and software-based firewalls), with the purpose of increasing both the effective bandwidth and the number of concurrently monitored connections. In this approach, the firewall is positioned at the network boundaries on an edge router line card and it is provided as a value-added service to ISPs or subscribers. In this manner, the enterprise security zone extends beyond boundaries of the enterprise itself, and is implemented within the premises of the service provider.

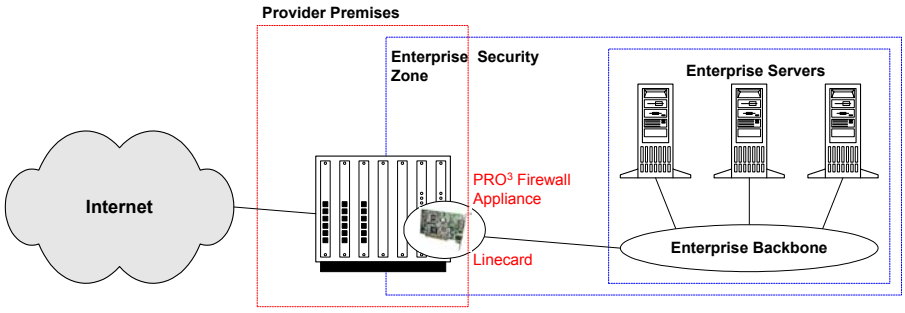

Fig. 3. Firewall physical configuration

The PRO^3 system integrated on the line card, aims to accelerate the performance of the firewall by implementing key functionality in hardware, as well as optimizing the balance between hardware and software functions. It is expected that significant performance enhancements can be achieved regarding:

- The rate of connection insertion/deletion, which measures the number of connections per second supported by the system, when applications create and destroy connections continuously.
- The throughput, which measures the aggregate number of bytes per packet that the system can process and forward to its output interface.

- The latency, which measures the aggregate delay encountered by network traffic and which is introduced by the processing delay of the system.
- The number of concurrent connections, which measures the maximum number of simultaneous connections supported by the system.

4.1 Monitoring of streams of packets on the network layer

The protocol processing capabilities of PRO3 extend from the network layer up to the application layer. Integrated in a firewall system the PRO3 processor shall extract all necessary protocol and state information from the IP layer and use it for higher layer processing. The actual protocol processing required on the IP layer apart from the packet forwarding functionality, which makes use of some header values in order to classify and forward the packet, extends to functions regarding handling of fragmented packets and reassembly. Processing on the IP layer for security applications require classification of packets into their corresponding flows, monitoring of the packet stream per flow and handling of fragmented IP packets.

There are several security considerations regarding fragmented IP packets entering a network. Packets that are bigger than the maximum size the underlying layer can handle (the MTU) are fragmented into smaller packets, which are then reassembled by the receiver. Fragmented packets with deliberately falsified application/protocol information data length may cause buffer overflows on some systems, when the fragments are reassembled at the other end into a complete packet (e.g. a ping application attempting to transmit more than 65507 octets of data will create an ICMP packet larger then the maximum IP data length of one packet, hence it will be transmitted over multiple IP fragments and may cause an overflow of 16-bit internal variables on reassembly, depending on the implementation of the receiver).

To cope with such illegal packets a firewall could provide the functionality either to entirely block fragmented IP packets or forward them according to specific programmable rules. Forwarding based on programmable rules may include monitoring of transport layer sessions and forwarding of fragmented IP packets when they carry legal information according to the session's state. Complete or partial termination of the IP layer is also required in some cases of packet processing on higher layers e.g. TCP packet reassembly and/or application layer monitoring/termination.

4.2 Monitoring of transport layer sessions

The Transport Control Protocol (TCP) is a connection-oriented, reliable transport protocol. For the communication of two hosts, a TCP connection must first be established before data may be exchanged. Applications using the TCP protocol exchange data through *sockets*. Sockets represent the TCP's connection endpoints and are defined as a 4-tuple: the IP source/destination address and the TCP source/destination port. Thus information from the IP header as well as from the TCP header is used for the connection lookup. TCP uses a three-way handshake between the two hosts for the connection establishment, assigns sequence numbers to every byte in every segment, within the advertised window sizes and acknowledges all data bytes received

from the other end. Malicious users usually attempt to obscure normal network operation and system break-down using packets with falsified protocol information so as to cause a system to stop responding or crash (Denial of Service-DoS attacks).

Firewalls managing state information for TCP applications must maintain a state table registering the transaction of information for each session and accepting packets only when they are conforming to the TCP state transitions. For a session to be registered as established and packets from this connection being let through the Firewall the three-way handshake must be monitored and complete. The flow of packets belonging to legitimate and established managed connections may also be monitored for consistent state information e.g. sequence numbers that lie within the anticipated range (compared to the sequence number of the previous packet, the announced window size from the receiver etc.). The protocol information required for managing a connection's state and the complete set of connection's state parameters that may be registered per connection are included in Table 1. Each entry in the state table should be managed for a configurable time-interval. Configurable counters/timers should indicate time-out events for idle connections i.e. connections for which no packet has been logged within the configured time-interval.

Table 1. TCP connection state management parameters

Protocol Values	Session managed parameters
<IP source>	TCP State
<IP destination>	Sender SN
<Protocol>	Receiver SN
<TCP source port>	Sender WIN
<TCP destination port>	Receiver WIN
<TCP Flags>	

A half-duplex implementation of a firewall could also provide a reduced capability for management of session state information. In this case a flow of packets could be monitored only in the one direction (originating from a network side considered insecure towards a network segment to be secured). Rules for accepting packets could then be based on traffic and overall network statistics information (e.g. number of open connections, rate of increase of new connections etc.)

UDP is a connectionless protocol, thus there is no explicit definition of UDP flows. Applications using the UDP protocol though use the same communication mechanism through sockets though as TCP. Implicitly since each application data transaction lasts for a certain time interval, there can be an approximation of UDP "connections" for a configurable time interval, when packets between two specific UDP endpoints are logged. Restrictions imposed according to statistics of such approximated UDP connections may indicate illegal network traffic and usage.

Configurable thresholds and timeout events for both TCP and UDP connections may be used in a Firewall to assure the secure operation of a network. Certain timers may be used per connection as well as overall statistics for established connections, half-open TCP connections and rate of increase of connections must be maintained. Table 2 includes an indicative list of timers to be managed per session and Table 3 contains a list of statistics and threshold values to be maintained by a Firewall managing session state and overall network state information.

Table 2. Indicative list of timers to be managed per session

Timeout Value	Range
Length of time the software waits for a TCP session to reach the established state before dropping the session.	seconds-minutes
Length of time a TCP session will still be managed after the firewall detects a FIN-exchange.	seconds
Length of time a TCP session will still be managed after no activity	minutes-hours
Length of time a UDP session will still be managed after no activity	seconds-minutes
Length of time a configured application session will still be managed after no activity.	Seconds

Table 3. Indicative list of statistics and threshold values to be maintained

Threshold Value	Range
Number of allowed existing half-open sessions.	Thousands
Allowed rate of new un-established sessions.	hundreds-thousands
Number of existing half-open TCP sessions per destination host address.	Hundreds

4.3 Monitoring of application layer sessions

The TCP and UDP connections as described above are tied to the higher layer protocol/application using these transport layer protocols for data transmission. Monitoring of connection state and packet information is related to higher layer application message transactions and awareness of higher layer protocols structure is a required feature in packet filtering applications that function on the application layer. Since the TCP/IP suite commits specific port numbers for specific applications information of the TCP and UDP header is used to directly indicate the application and the context of the TCP or UDP data. Monitoring of TCP/UDP data and management of application layer sessions could also be implemented in a firewall environment. The PRO^3 processor in this case shall extract specific protocol information from the TCP/UDP data executing specific code for each configured application.

A number of example applications that could be managed by a Firewall system using the PRO^3 processor are TELNET [10], FTP [11], SNMP [12], HTTP [13, 14], H.323. The list of potential applications could extend further since any existing or future application could be processed with the appropriate code implementation and reprogramming of the PRO^3 processor.

4.4 Line card architecture

The boundaries of operation of a stateful inspection firewall have not been completely defined. Implementations are based upon the required level of security and range from simple monitoring of TCP messages exchanged between two networks to the more sophisticated applications state monitoring of the protected network. PRO^3

provides the capability of achieving any level of security, either by operating in a standalone mode, or through its co-operation with an external processor.

The demonstration system will be a bridge positioned between a corporate LAN and the Internet. A board incorporating PRO^3 will be developed providing two high speed links (OC-48) one to the LAN and the other to the service provider. Furthermore, an amount of fast memory will be integrated, while an interface to an external processor will also be provided. The described system is shown in Fig.4.

The board will demonstrate the capability of the PRO^3 system to accelerate TCP/IP functions utilized in firewalling applications. In this manner it will be shown that efficient partitioning of traditionally software implemented functions between hardware and software, which will be running on a customized RISC processor, will provide the capability to increase the performance of an advanced firewall.

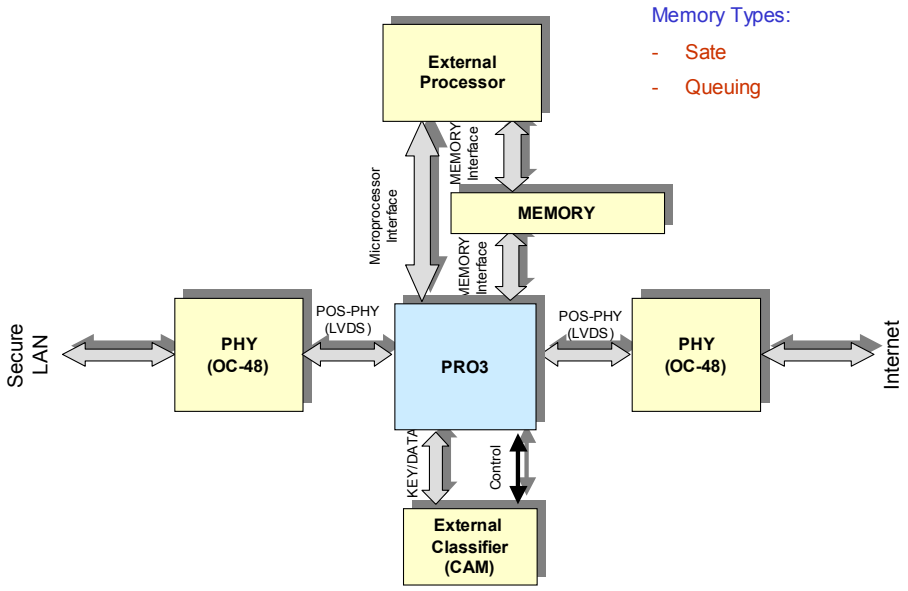

Fig. 4. Architecture of stateful inspection firewall

8. Conclusions

The most important push behind the network and protocol processor industry is bandwidth. Currently, due to fiber optics and other advances in the data transport medium, it is possible to transmit more information than a CPU can process. Additionally, with Terabit technology on the horizon, the data transmission speed will increase another 100 times faster than OC192. With this kind of bandwidth and speed

looming in the future, the network and protocol processor was developed and continues to be refined for optimized networking applications.

The PRO^3 system will advance the state-of-the-art in and will introduce a new concept in high-performance processor architectures by integrating a RISC and a RPM. Furthermore, the processor will open a new roadmap in developing special processors able to support a large number of networking applications. It is expected that the processor will accelerate considerably the execution of protocols thus allowing the realization of advanced and cost effective telecommunication applications.

References

1. D. Liu, U. Nordqvist, C. Svensson, "Configuration based architecture for high speed and general purpose protocol processing," IEEE Workshop on Signal Processing Systems, October 1999.
2. D. Niehaus, A. Battou, A. McFaralnd, B. Decina, H. Dardy, V. Sirkay, B. Edwards, "Performance benchmarking of signalling in ATM networks," IEEE Comm. Mag., August 1997.
3. C. Maeda, B. Bershad "Protocol service decomposition for high performance networks," Proceedings of he 14th ACM Symposium on Operating Systems Principles, December, 1993.
4. D. Feldmeier, A. McAuley, J. Smith, D. Bakin, W. Marcus, T. Raleigh, "Protocol boosters," IEEE Journal on Selected Areas in Comm., vol. 16, No. 3, April 1998.
5. PRO^3, IST project 99-11449 presentation, http://www.cordis.lu/ist/projects/99-11449.htm.
6. Intel, IXP1200 Network Processor, http://developer.intel.com/design/network/IXP1200.htm
7. Motorola, C-Port, http://www.cportcorp.com/.
8. Solidum Systems Corp., PAX.port 1100, www.solidium.com.
9. Agere, NPFPP, NPRSP, NPASI, FPL, IDS, http://www.agere.com.
10. J. Postel, J. Reynolds, "Telnet Protocol Specification", RFC 854, May 1983
11. J. Postel, J. Reynolds, "File Transfer Protocol (FTP)", RFC 959, October 1985
12. J. Postel , "Simple Mail Transfer Protocol", RFC 821, August 1982
13. T.Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996
14. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997
15. ITU-T Recommendation H.323 "Packet-Based Multimedia Communications Systems".